AMENDMENTS TO THE CLAIMS

This listing of the claims will replace all prior versions, listings, of claims in the application:

Claims 1-9 (canceled)

Claim 10 (currently amended): A method for use with commands sent to a network communications coupler by at least one client on a network connected to said coupler comprising ~~the steps of~~:

designating by said at least one client by means of a predetermined key value provided by said at least one client which of said commands are delayed response commands, said at least one client using said predetermined key value to retrieve a reply to said at least one delayed response command executed by said coupler;

A11

storing in said coupler a first instance of a template, said first instance used to store in said coupler at least one of said delayed response commands during execution of said at least one delayed response command by said coupler; and

storing in said coupler a second instance of said template, said second instance used to store in said coupler a reply to said at least one delayed response command executed by said coupler until said reply is retrieved by said client.

Claim 11 (original): The method of Claim 10 wherein said template has a declaration having one template parameter class in the parameter list of said template and one or more statements defining said template.

Claim 12 (original): The method of Claim 11 wherein said one or more statements defining said template include a pointer of said template parameter class.

Claim 13 (original): The method of Claim 12 wherein said one or more statements defining said template further include enumerated special values for designating boundary and special conditions and one or more methods.

Claim 14 (original): The method of Claim 13 wherein said one or more methods statements comprise constructor, destructor and access methods.

Claim 15 (currently amended): The method of Claim 11 wherein said template is:

```
template <class Entry> KeyedStorage<Entry>::KeyedStorage()
{
   storage = new Entry [1+MAXKEYVAL];

   for( int i = 0; i <= MAXKEYVAL; i++)
   {
      storage[i] = NULL;
   }
};

template <class Entry> KeyedStorage<Entry>::~KeyedStorage()
{
   delete[] storage;
};

template <class Entry> int KeyedStorage<Entry>::putEntry( int
key, Entry newEntry )
{
   int success;

   if( key >=0 && key <= MAXKEYVAL )
   {
      storage[key] = newEntry;
      success = TRUE;
   }
   else
   {
      success = FALSE;
   }

   return( success );
};

template <class Entry> Entry KeyedStorage<Entry>::getEntry( int
key )
{
   Entry foundEntry;

   if( key >= 0 && key <= MAXKEYVAL )
   {
      foundEntry = storage[key];
      storage[key] = NULL;
   }
```

A11

```
   else
   {
      foundEntry = NULL;
   }

   return( foundEntry );
};

template <class Entry> Entry KeyedStorage<Entry>::checkEntry(
int key )
{
   Entry foundEntry;

   if( key >= 0 && key <= MAXKEYVAL )
   {
      foundEntry = storage[key];
   }
   else
   {
      foundEntry = NULL;
   }

   return( foundEntry );
};

int KSTest(void )
{
   KeyedStorage<CNIReply *> ReplyStorage;
   KeyedStorage<CNICommand *> ActiveCommands;
   KeyedStorage<ClientInterface *> ClientInterfaces;

   CNIReply * Reply1 = (CNIReply *)1;
   CNIReply * Reply2 = (CNIReply *)2;

   CNICommand *Command1 = (CNICommand *)101;
   CNICommand *Command2 = (CNICommand *)102;

   ClientInterface *Client1 = (ClientInterface *)1001;
   ClientInterface *Client2 = (ClientInterface *)1002;

   ReplyStorage.putEntry( 1, Reply1 );
   ActiveCommands.putEntry( 1, Command1 );
   ClientInterfaces.putEntry( 1, Client1 );

   ReplyStorage.putEntry( 2, Reply2 );
   ActiveCommands.putEntry( 2, Command2 );
   ClientInterfaces.putEntry( 2, Client2 );

   printf("\n\ralles put\n\r");

   printf("\n\rchecking first reply = %p",
ReplyStorage.checkEntry( 1 ) );
```

A11

```
   printf("\n\rchecking second reply = %p",
ReplyStorage.checkEntry( 2 ) );

   printf("\n\rchecking first command = %p",
ActiveCommands.checkEntry( 1 ) );
   printf("\n\rchecking second command = %p",
ActiveCommands.checkEntry( 2 ) );

   printf("\n\rchecking first client = %p",
ClientInterfaces.checkEntry( 1 ) );
   printf("\n\rchecking second client = %p",
ClientInterfaces.checkEntry( 2 ) );

   printf("\n\rgetting first reply = %p", ReplyStorage.getEntry(
1 ) );
   printf("\n\rgetting first reply again = %p",
ReplyStorage.getEntry( 1 ) );

   printf("\n\rgetting second command = %p",
ActiveCommands.getEntry( 2 ) );
   printf("\n\rgetting second command again = %p",
ActiveCommands.getEntry( 2 ) );

   printf("\n\rgetting first client = %p",
ClientInterfaces.getEntry( 1 ) );
   printf("\n\rgetting first client again = %p",
ClientInterfaces.getEntry( 1 ) );

   printf("\n\r");

   return 0;
}
```

A11

Claim 16 (currently amended): The method of Claim 10 further comprising ~~the step of~~ defining for said template a declaration having one template parameter class in the parameter list of said template and one or more statements defining said template.

Claim 17 (currently amended): The method of Claim 16 further comprising ~~the step of~~ defining for said one or statements a pointer of said one template parameter class, enumerated special values for designating boundary and special conditions, and one or more methods.

Claim 18 (currently amended): The method of 17 further comprising ~~the step of~~ defining for said one or statements that are method statements methods selected from constructor,

destructor and access methods.

Claim 19 (new): A system comprising:

a first network having at least one client connected thereto;

a second network;

said first and second networks having different protocols;

a network communications coupler connected to both said first network and said second network;

said at least one client sending a plurality of commands to said coupler and designating at least one of said plurality of commands as a delayed response command;

said coupler for storing a first instance of a template, said first instance used to store in said coupler at least one of said delayed response commands during execution of said at least one delayed response command by said coupler; and for storing a second instance of said template, said second instance used to store in said coupler a reply to said at least one delayed response command executed by said coupler until said reply is retrieved by said client.

A11

Claim 20 (new): The system of claim 19 further comprising a distributed control system connected to said second network.

Claim 21 (new): The system of claim 19 wherein said at least one client designates by means of a predetermined key provided by said at least one client which of said plurality of commands sent to said coupler are said delayed response commands, said at least one client using said predetermined key value to retrieve a reply to said at least one delayed response command executed by said coupler.

Claim 22 (new): A system comprising:

a first network having at least one client connected thereto;

a distributed control system comprising a second network;

said first and second networks having different protocols;

a network communications coupler connected to both said

first network and said second network;

said at least one client sending a plurality of commands to said coupler and designating by means of a predetermined key value provided by said at least one client at least one of said plurality of commands as a delayed response command, said at least one client using said predetermined key value to retrieve a reply to said at least one delayed response command executed by said coupler;

said coupler for storing a first instance of a template, said first instance used to store in said coupler at least one of said delayed response commands during execution of said at least one delayed response command by said coupler; and for storing a second instance of said template, said second instance used to store in said coupler a reply to said at least one delayed response command executed by said coupler until said reply is retrieved by said client.

A11